Eötvös Loránd University

Faculty of Informatics

Dept. of Data Science and Engineering

Predictive analysis in Healthcare: XGboost optimization with PSO and the Firefly Algorithm

Supervisor:
Molnár Bálint
Dr. Habil. Assistant Lecturer

Author:
Tshifaro Justin Ndivhuwo
Computer Science MSc

Budapest, 2023

Computer Science MSc
1st Semester, 2022-2023 Academic Year

Thesis Topic Description

Submission deadline: 2022.09.30.

Student's Data:

Name: Tshifaro Ndivhuwo Justin
Neptun Code: SUQG0J

Course Data:

Student's Major: Computer Science MSc
Semester & Academic Year of Student's Admission: 2021/22/1

A brief description of the thesis topic:

(Consulting with your potential supervisor, give a synopsis of your planned thesis in a few sentences.)

Predictive analysis in Health care
The goal of predictive analytics is to determine the likelihood of future outcomes based on historical data. It does this by using data, statistical algorithms, and machine-learning approaches. The objective is to provide the best prediction of what will occur in the future, rather than simply knowing what has already occurred.
Predicting the future with the aid of machine learning algorithms is no longer a challenging task, particularly in the field of medicine where it is now possible to predict diseases and anticipate a cure. An overview of the development of big data in the healthcare system will be provided in this paper, and a learning algorithm will be applied to a set of health data. This dissertation aims to predict chronic diseases and mental health using Clustering and Classification models. Both Disadvantages and Advantages Disadvantages will be discussed.

During the experimental design and development, adequate methodologies are used (Design Science Research, Software Case Study, Data Science data and system modeling, etc.). The developed model will be verified and validated by domain-specific metrics.

Thesis Supervisor: Dr. Bálint Molnár

Specialisation:
Data Science

Subject related to the topic of the thesis:

Intorduction to Data Science

Advanced Machine Learning

Open-source Technologies for Real-time Data Analytics

Contents

1 Introduction 1

1.1 Motivation and Objectives . . . 2

1.2 Literature Review . . . 3

1.2.1 Predictive Analysis in Healthcare . . . 3

1.2.2 XGboost and Swarm Intelligence . . . 4

2 Background and Problem Understanding 7

2.1 Machine Learning . . . 7

2.1.1 Types of Machine Learning . . . 7

2.1.2 Ensemble Methods in Machine Learning . . . 9

2.2 Machine Learning in Healthcare . . . 11

3 Methodology 12

3.1 Classification model . . . 12

3.1.1 XGBoost algorithm . . . 12

3.2 Optimizing Algorithms . . . 17

3.2.1 Particle Swarm Optimization (PSO) . . . 17

3.2.2 Firefly Algorithm . . . 20

3.3 Evaluation metrics: Classification Report . . . 24

4 Experiments and Results 26

5 Conclusion 27

Bibliography 28

Abstract

Chapter 1

Introduction

The healthcare industry is one of the most important sectors in the world, with the primary goal of delivering the best possible treatment to patients. With the advancement of data analytics and machine learning, predictive analysis has become an important element of healthcare[1], allowing valuable insights to be extracted from massive amounts of medical data. Predictive analytics techniques rely on historical data to create models that can forecast future results, allowing healthcare providers to make more educated decisions about diagnosis, treatment, and patient care [2].

Because of the broad implementation of electronic health records and other digital health technologies, the volume of healthcare data created has increased rapidly in recent years. The healthcare sector has both opportunities and challenges due to this massive amount of data. On the one hand, data availability presents enormous opportunities for enhancing patient outcomes and healthcare operations. On the contrary, the enormous amount and complexity of healthcare data make extracting valuable insights from it difficult. Predictive analytics is an effective technique for assisting healthcare workers in making sense of this massive volume of healthcare data. Predictive models can find patterns and predict future outcomes by evaluating previous data. Predictive models have many uses in healthcare, including disease outbreak prediction, identifying people at risk of developing particular illnesses and assessing therapy effectiveness[3].

XGBoost is a well-known machine learning method that is widely utilized in healthcare predictive modeling due to its accuracy, speed, and scalability [4]. XGBoost, like any other algorithm, has limitations, and its performance can be enhanced further by optimization techniques. Particle Swarm Optimization (PSO) and the Firefly Algorithm are two swarm intelligence-based optimization algorithms that have shown significant potential in improving the performance of machine learning systems [5]

Swarm intelligence is a metaheuristic optimization technique that solves complicated optimization problems by simulating the collective behavior of social creatures such as ants, bees, and birds. PSO and the Firefly Algorithm, both inspired by the behavior of fireflies and birds, are capable of traversing the search space more efficiently than classic optimization methods [6]. We aim to improve the predicted accuracy of the XGBoost algorithm for healthcare applications by using the power of swarm intelligence.

1.1 Motivation and Objectives

The XGBoost algorithm has demonstrated exceptional performance in predictive modeling applications, making it a preferred option for predictive analysis in the healthcare industry. Optimizing the hyperparameters of the XGBoost algorithm can, however, substantially boost the method's performance. Application of optimization techniques like PSO and Firefly to the XGBoost algorithm in the healthcare industry might result in large increases in predicted accuracy. These techniques have shown great promise in improving the performance of machine learning models.

The primary goal of this thesis is to investigate the potential of optimization techniques to improve the XGBoost algorithm's performance for healthcare predictive modeling. We seek to increase the precision and accuracy of the predictive models by applying PSO and Firefly methods to optimize the hyperparameters of the XGBoost algorithm. The research presented in this thesis will add to the expanding body of knowledge on predictive analytics in healthcare and offer insightful information on the potential of optimization methods to improve the efficacy of machine learning algorithms. In the end, this study attempts to assist medical practitioners in making knowledgeable choices regarding patient care, diagnosis, and treatment.

1.2 Literature Review

The research articles in this collection concentrate on using machine learning and predictive analytics in healthcare systems.

1.2.1 Predictive Analysis in Healthcare

The majority of the authors in this section created a framework or paradigm for using big data analytics in healthcare.

Filipe da Silva Gonçalves [7] outlines in their work "Predictive Analysis in Healthcare" a prototype predictive analytics algorithm for predicting waiting times in emergency departments. To create the most realistic model, the author takes real ED data from a Portuguese hospital and supplements it with external data such as weather information, DGS Announcements, and the number of football games played. The author uses the Nave Bayes and Random Forest algorithms in three different scenarios and indicates that the external data attributes added were not the most essential variables for waiting times, with triage colors and disease category being the most critical determinants.

Basma Boukenze1, Hajar Mousannif, and Abdelkrim Haqiq[8] in their paper "Predictive Analytics in Healthcare System Using Data Mining Techniques" suggest using the C4.5 decision tree method to predict chronic kidney disease, outperforming other classification algorithms and attaining high accuracy rates. The authors also emphasized that predicting the future is no longer a difficult endeavor because of the promises of predictive analytics in big data and the application of machine learning algorithms, notably for medicine because disease prediction and expecting a remedy became possible.

Sheng et al [9] offers a deep learning-based strategy to identify critical complication characteristics in patients with acute illnesses in paper "Predictive Analytics for Care and Management of Patients With Acute Diseases: Deep Learning–Based Method to Predict Crucial Complication Phenotypes" outperforming three benchmark methodologies assessed, using a dataset From a major health care organization in Taiwan with 10,354 samples of electronic health records that pertained to 6545 patients with peritonitis.

N Divyashree et al [10] describe the implementation of the LWGMK-NN multistratified method for improved clinical diagnosis, which outperforms nine state-of-the-art classification algorithms. The author made use of Ten common clinical datasets are used to compare the proposed work's performance to nine state-of-the-art classification algorithms: Logistic Regression, Decision Trees, Gaussian Naive Bayes, Random Forest, and Linear Support Vector Machine are all examples of machine learning techniques.

1.2.2 XGboost and Swarm Intelligence

In this section, most of the authors demonstrated and focused on the optimization of a predictive model optimized using swarm intelligence. Mostly Focusing on XGBoost, Particle swarm optimization, and Firefly Algorithm. In "A Swarmbased Optimization of the XGBoost Parameters" paper, Ali Haidar et al [11] suggested a novel method for choosing the XGBoost algorithm's parameters that makes use of the Particle Swarm Optimization (PSO) algorithm. The PSO algorithm's characteristics are examined and analyzed by the authors, who demonstrate that the number of iterations has a stronger influence on model performance than the number of particles. The suggested method is practical for choosing machine learning algorithm parameters, particularly when resources like time and infrastructure are scarce. The outcomes suggest that PSO may be used to determine the best options in parameter selection tasks and that XGBoost combined with PSO can be a powerful forecasting technique. An additional study will examine new optimization algorithms and analyze the impact of other PSO attributes.

Chao Qin et al [12] suggested a credit scoring model based on the XGBoost algorithm that has been APSO-optimized. Using five widely used assessment metrics, the model is assessed on four credit datasets and seven KEEL benchmark datasets. The findings demonstrate that the suggested model performs better than other models on average and that APSO outperforms alternative hyperparameter optimization techniques. For the majority of datasets, the APSO-XGBoost model achieves the highest prediction accuracy, F1-score, and probability prediction capabilities. According to the study, APSO can enhance model accuracy by encouraging the XGBoost structure's alignment with the properties of the credit data and overcoming the particle swarm's local optimality.

Miodrag Zivkovic et al [13] suggested a novel improved version of the firefly algorithm, called CFAEE-SCA, for hyper-parameter optimization of XGBoost classifiers for network intrusion detection in the article "Novel hybrid firefly algorithm: an application to enhance XGBoost tuning for intrusion detection classification." On benchmark instances, the CFAEE-SCA algorithm is compared to other cutting-edge metaheuristics and found to have a lot of promise for overcoming machine learning hyper-parameter optimization problems. It is also demonstrated that the CFAEE-SCA-XGBoost strategy works much better than the standard XGBoost method and outperforms other metaheuristic approaches for network intrusion detection.

Using a reduced risk factor list, three ensemble-based classification approaches (Random Forest, Extreme Gradient Boosting, and AdaBoost), and the Firefly optimization algorithm, Irfan Ullah Khanand Peña, Antonio J. et al [14] suggests a study for the early identification of cervical cancer. Accuracy, sensitivity, specificity, positive predictive accuracy (PPA), and negative predictive accuracy (NPA) are used to assess the effectiveness of the suggested model. The work uses the Synthetic Minority Oversampling Technique (SMOTE) with the Cervical cancer Risk variables data collection, which contains 32 risk variables and four targets (Hinselmann, Schiller, Cytology, and Biopsy). When the proposed models are used to diagnose cervical cancer using data from reduced risk factor studies as a baseline, they perform noticeably better in terms of accuracy.

Using an extreme gradient boosting machine learning model optimized by a modified version of the multi-verse optimizer metaheuristic, Aleksandar Petrovic et al,[15] provides a framework for intrusion detection systems. On the UNSW-NB intrusion dataset, the suggested solution performed better than other cutting-edge methods, resulting in gains in device productivity, efficiency, and communication. However, the paper also draws attention to the security risks brought on by the concept of Industry 4.0's increased diversity and number of devices. In order to demonstrate the superiority of the suggested solution, the study creates a common evaluation system utilizing truthfulness and polarity units. Comparing the best, worst, and mean values obtained by the proposed XGBoost-IMVO model to those of other competing metaheuristics, XGBoost-MVO obtained the best median, and XGBoost-BA obtained the best Std and Var scores.

Overall, these studies show the potential of machine learning and predictive analytics in healthcare systems, particularly in terms of improving disease diagnosis and prognosis. These techniques provide a more accurate and effective approach to processing massive amounts of medical data and finding patterns that could be hidden from human experts. According to the findings of these studies, these strategies may result in better patient outcomes and more effective healthcare delivery. However, more investigation is required to assess how well these techniques perform in various healthcare contexts and to address issues with data privacy and ethical issues.

Chapter 2

Background and Problem Understanding

2.1 Machine Learning

Machine learning is a branch of artificial intelligence (AI) and computer science that focuses on using data and algorithms to mimic human learning processes and progressively increase accuracy [16].

2.1.1 Types of Machine Learning

Supervised Machine learning

Supervised learning is a sort of machine learning in which the output is predicted by the machines based on well-labeled training data. The term "labeled data" refers to input data that has already been assigned to the appropriate output. In this type of machine learning, the training data given to the machines act as the teacher who teaches the machines to correctly predict the output. It employs the same idea that a student would learn under a teacher's guidance [17]. Regression and Classification are the two types of supervised learning. classification Algorithms are used When an output variable has two classes, such as Yes-No, Male-Female, or another categorical outcome. Random Forest, Decision Trees, Logistic Regression, and Support Vector Machines are some of examples of classification. Regression algorithms are used when the input variable and the output variable are related, it is used to predict Continuous variable. Examples of regression algorithms include Bayesian linear regression, linear regression, regression trees, etc.

Unsupervised machine learning

Unsupervised learning is a type of machine learning where the users don't have to watch over the model. Instead, it enables the model to operate independently and find previously undetected patterns and information [18]. It is mostly used when we have unlabeled data. Unsupervised Machine Learning can further be classified into Clustering (e.g. K-means clustering. KNN (k-nearest neighbors) and Association (e.g. Apriori algorithm)

Semi-supervised learning

Semi-supervised learning is a type of Machine learning that involves both labeled and unlabeled data which means it is neither supervised nor unsupervised, it is in between. It guides classification and feature extraction from a larger, unlabeled data set using a smaller, labeled data set. examples of this type of learning are text document classifiers, Web content classification, and Speech recognition

Reinforcement machine learning

Reinforcement machine learning is a machine learning model similar to supervised learning, except that the algorithm is not trained on sample data. This model learns as it goes along through trial and error[16]. For each positive action, the agent receives positive feedback while for each negative action, the agent receives negative feedback or a penalty. It is mostly used in Automated Robots, Natural Language Processing (NLP), Image processing, etc.

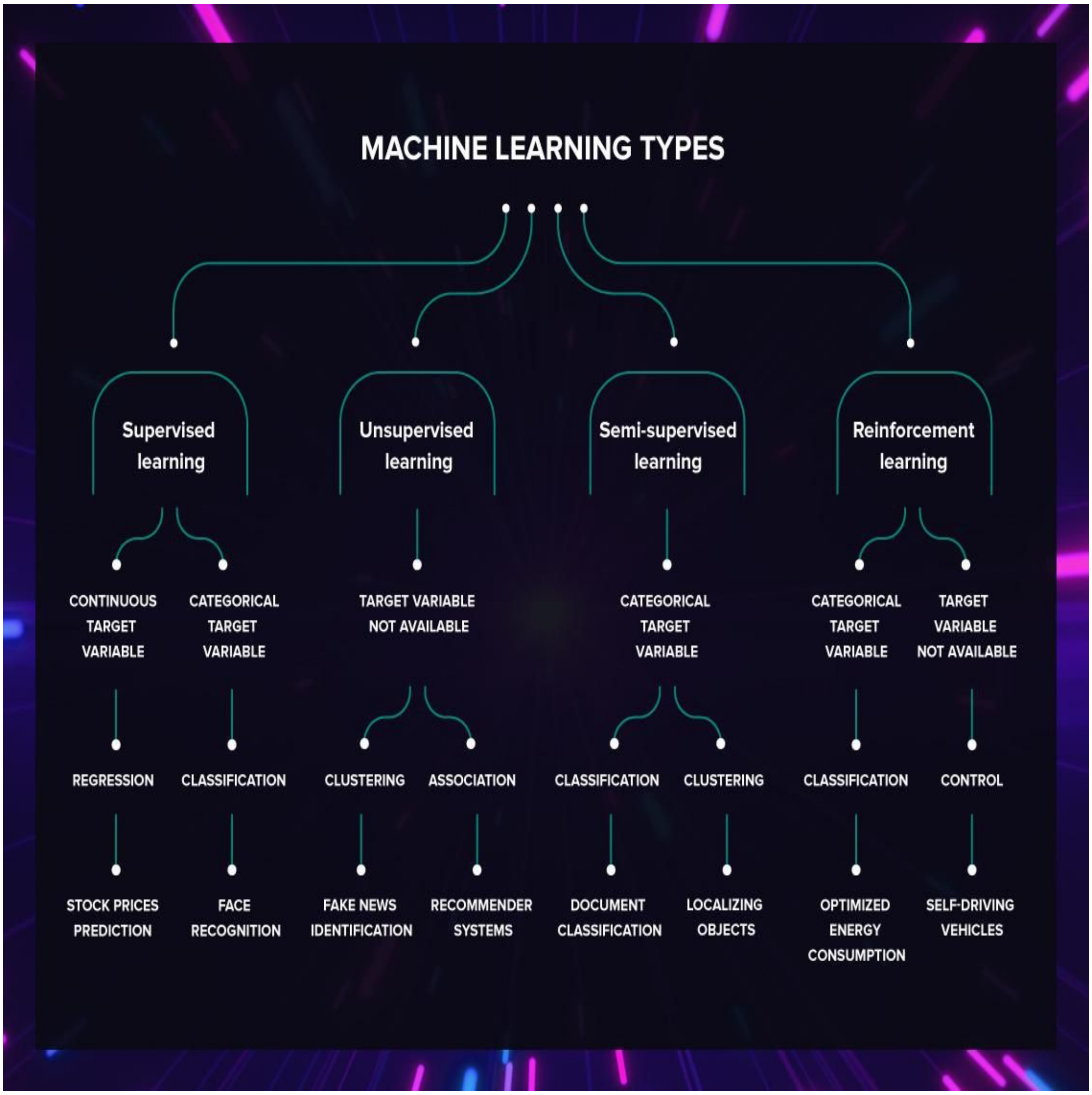

Figure 2.1: Types of Machine learning and use-case examples
[19]

2.1.2 Ensemble Methods in Machine Learning

Ensemble techniques are a type of machine learning technique that combines multiple base models to create a single best predictive model. Rather than building a single model and hoping it is the best or most accurate predictor possible, ensemble approaches consider a variety of models and average them to get a single final model. The core idea behind ensemble learning is to reduce errors or biases in individual models by using the collective intelligence of numerous models, resulting in a more exact prediction. Ensemble Methods are one of the most powerful and easy to use predictive analytics techniques.

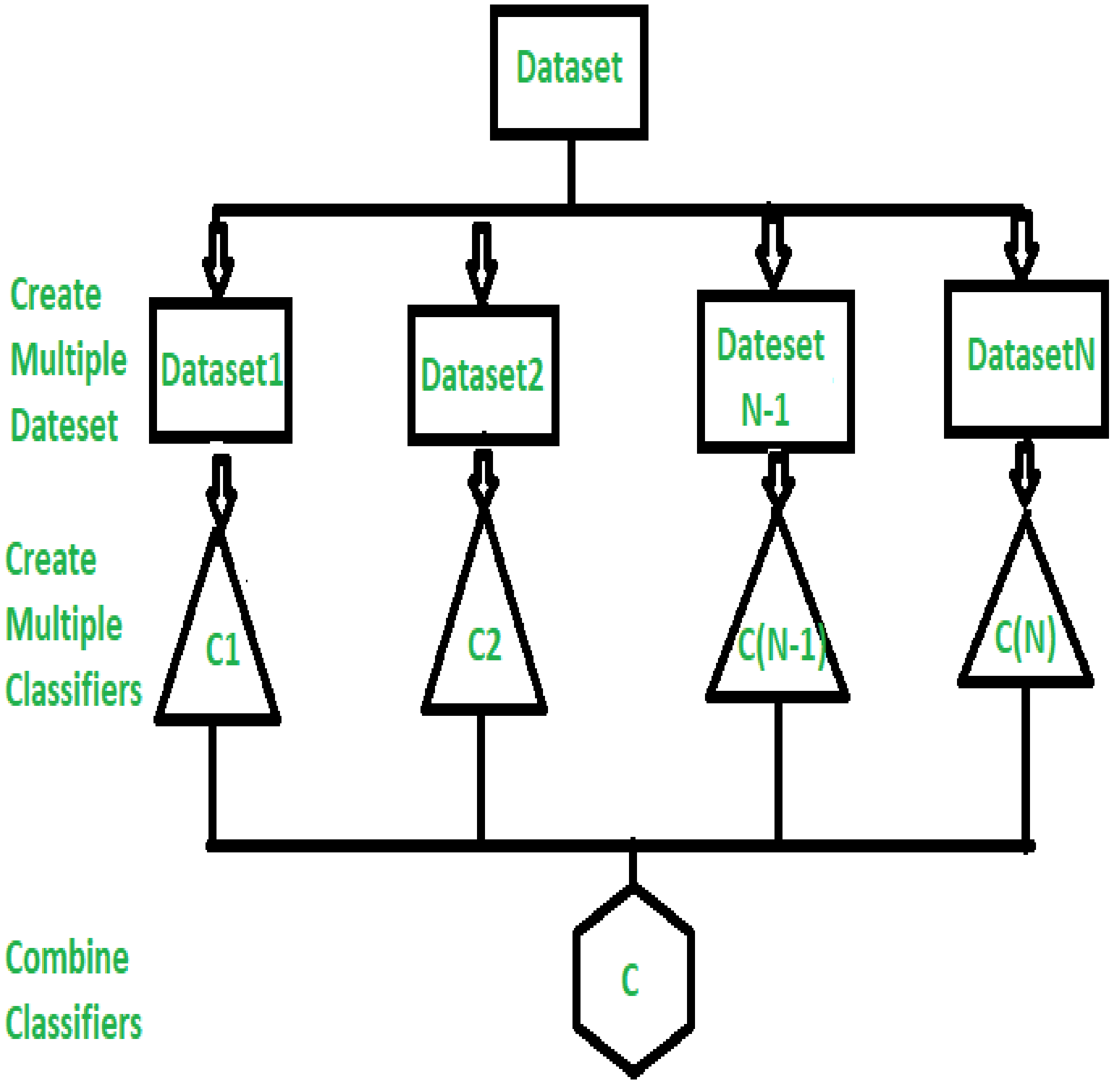

Figure 2.2: General structure of Ensemble method
[20]

The three main types of ensemble learning methods are:

- **Bagging** is an ensemble meta-estimator that applies base classifiers to separate, randomly selected subsets of the original dataset, then combines (either by voting or by averaging) each result into a final prediction. E.g Bagging meta-estimator and Random forest
- **Boosting** methods work by learning weak classifiers that are very marginally correlated with the true classification and then combining them to create a strong classifier that is highly connected with the genuine classification. E.G XGBoosr, AdaBoost, Light GBM, etc

- **Stacking** works by combining the predictions of numerous weak learners and Meta learners to produce a superior output prediction model

2.2 Machine Learning in Healthcare

Machine learning is a branch of artificial intelligence (AI) and computer science that focuses on using data and algorithms to mimic human learning processes and progressively increase accuracy [16]. Machine learning has become an important element of Data Science where Algorithms are trained using statistical techniques to produce classifications or predictions and to find important insights in data mining projects. One of the most crucial technologies in healthcare is machine learning, which aids medical personnel in managing clinical data and providing patient care by using predictive models

Chapter 3

Methodology

This chapter aims to describe and fully understand the research methods used. This research's primary objectives are to build a predictive machine learning model, train them on health-related datasets, and optimize its hyper-parameters utilizing swarm intelligence techniques, such as Particle Swarm Optimization (PSO), and Firefly Algorithm. The models and algorithms deployed for this project are thoroughly described below.

3.1 Classification model

3.1.1 XGBoost algorithm

Extreme Gradient Boosting, often known as XGBoost, is an efficient machine learning technique that is frequently applied in data science and machine learning applications. The Gradient Boosting Decision Tree (GBDT) method, on which it is based, is known for its effectiveness, scalability, and accuracy [21].

The XGBoost algorithm operates by assembling a group of decision trees. It creates decision trees iteratively and combines them to produce a final prediction. XGBoost detects sections of the data where the model performs badly in each iteration and concentrates on improving those areas in the next iteration. This process is repeated until the model reaches the appropriate level of precision.

XGBoost is popular among data scientists and machine learning practitioners due to a number of essential features. Its ability to handle missing data is one of its most crucial qualities. It can also handle both numerical and categorical data and can discover the optimum way to represent categorical variables in the model automatically.

XGBoost is also well-known for its scalability and efficiency. It is designed to take advantage of parallel processing and can handle massive datasets efficiently. XGBoost has also been tuned for performance, including a variety of strategies that reduce memory usage and calculation time.

The adoption of regularization techniques is one of the reasons why XGBoost is so accurate. These strategies prevent overfitting and improve model generalization. XGBoost additionally includes a set of hyper-parameters that can be tuned to improve the model's performance.

To understand the XGBoost well, it is crucial to understand the decision tree first. Figure 3.1 shows the structure of a Decision Tree and figure 3.1 Shows the structure of XGBoost algorithm

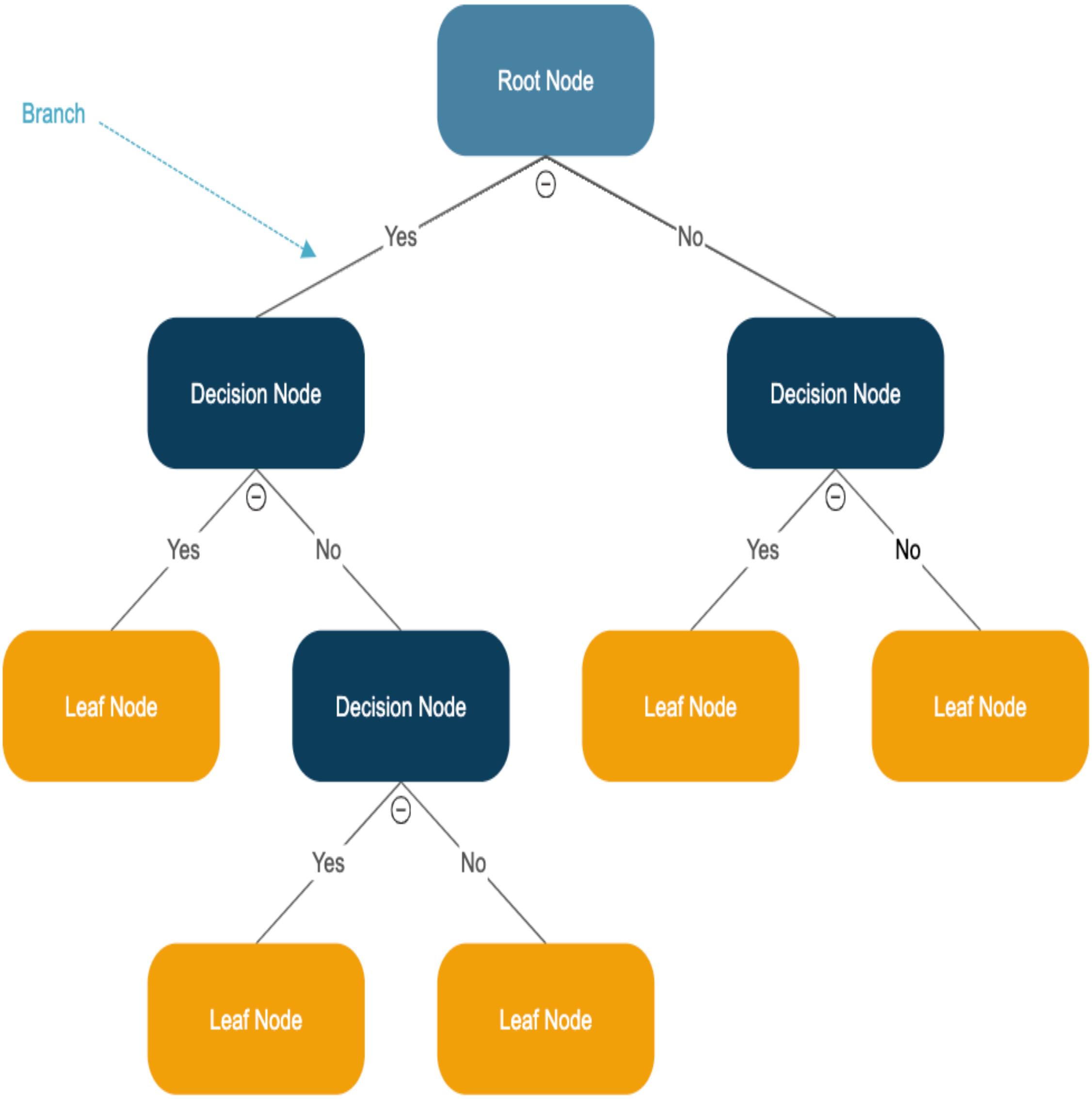

Figure 3.1: Decision Tree Structure
[22]

XGBoost Hyper-parameters

A hyperparameter is a machine learning parameter whose value is determined before training a learning algorithm.

1. learning_rate This influences the size of the step for each iteration of the boosting procedure. Lower values will make the model more conservative but converge may take longer iterations. Smaller learning rates generally require more trees to be added to the model. It is often referred to as the "shrinkage factor," and it is typically set to a low number, such as 0.1 or 0.01, to guarantee that the model converges slowly and smoothly.

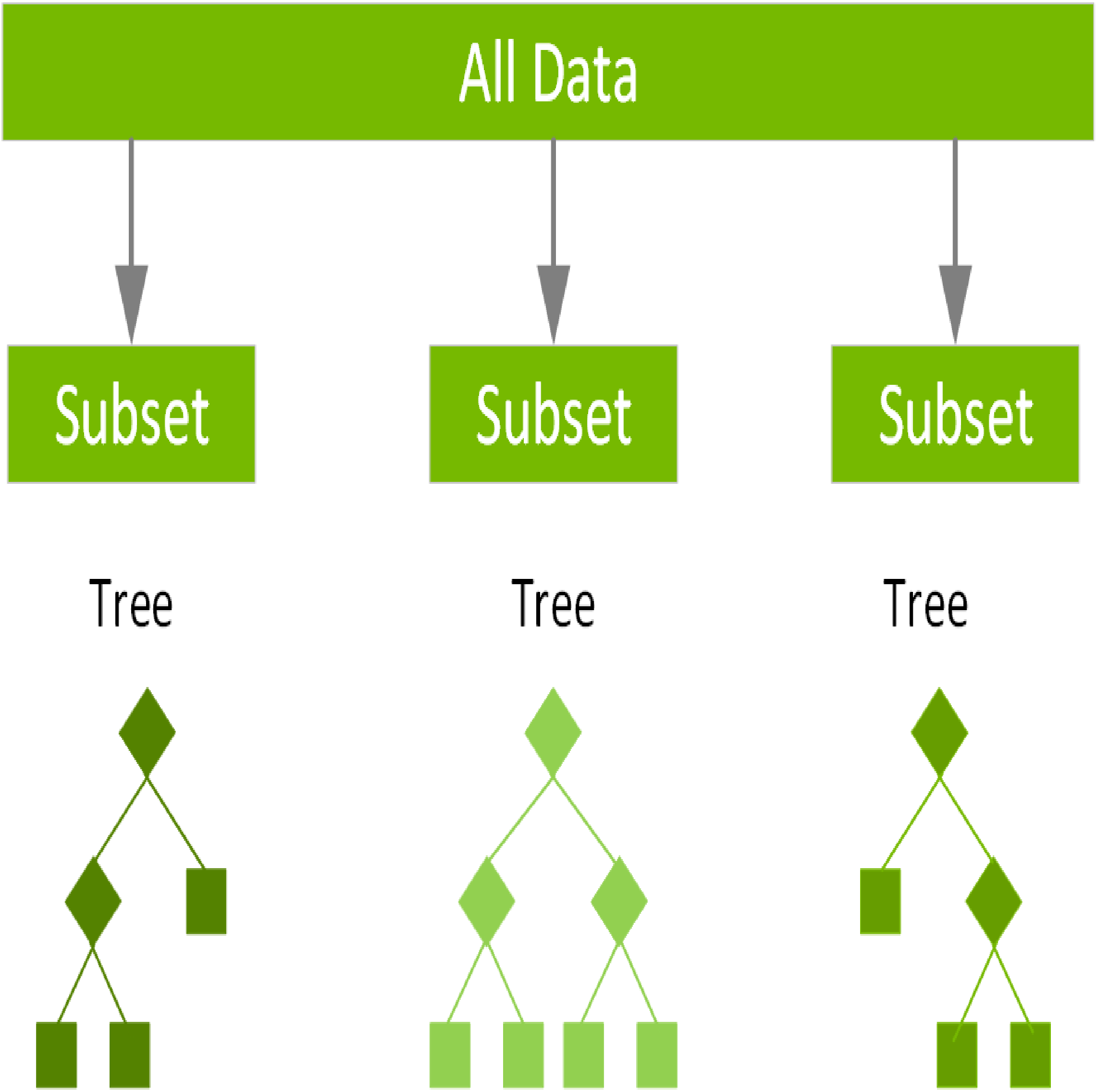

Figure 3.2: Decision Tree Structure
[23]

2. max_depth This regulates each tree in the ensemble's maximum depth. However, deeper trees run the risk of overfitting the data and capturing more complicated interactions. In simple words, it controls the size of decision trees.

3. Subsample This regulates how much of the training data is utilized to create each tree. Smaller numbers will increase the model's resistance to noise, but they can compromise its capacity to detect complex trends in the data. Greater and more complicated models are produced by greater subsample values, which might result in overfitting. This value is typically set between 0.5 and 1.

4. colsample_bytree The colsample_bytree parameter determines how many fea-

tures are used for each tree. A lower colsample_bytree number produces smaller and less complex models, which can aid in the prevention of overfitting. A higher colsample_bytree value leads to larger and more complicated models, which can result in overfitting.

5. n_estimators This determines how many trees are in the ensemble. More trees can improve model accuracy while also increasing the danger of overfitting.

6. min_child_weight This specifies the minimum instance weight required to split a node. Higher values will reduce overfitting and make the model more conservative.

7. gamma This regulates the minimal loss function decrease necessary to split a node. Higher values will reduce overfitting and make the model more conservative.

These are just a few of the many parameters that may be tweaked in XGBoost. There are numerous hyper-parameters, but selecting the optimal mix of parameters for a specific problem takes rigorous experimentation and adjustment, and for these experiments, we chose the hyper-parameters listed above. Table 3.1 shows the recommended and default value of XGBoost Hyper-parameters.It should be noted that these recommended values are not always ideal for every dataset and problem and that a hyperparameter tuning process is recommended to determine the best values for your unique instance.

Hyper-parameter	Default Value	Recommended Range
n_estimators	100	100 – 1000
max_depth	6	3 – 10
learning_rate	0.3	0.01 – 0.2
min_child_weight	1	1 – 10
subsample	1	0.5 – 1
colsample_bytree	1	0.5 – 1
gamma	0	0 – 5

Table 3.1: XGBoost default Hyper-parameters

3.2 Optimizing Algorithms

3.2.1 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a metaheuristic optimization technique inspired by the social behavior of birds and fish. Kennedy and Eberhart first proposed it in 1995. PSO is a population-based optimization technique that finds the global optimum by simulating the behavior of particles in a multidimensional search space[24].

PSO Concept

PSO starts a set of particles at random in the search space, and each particle represents a possible solution to the optimization issue. Each particle moves across the search space by modifying its position based on its own and its neighbors' experiences. Each particle's position is iteratively updated, and the quality of each particle's solution is evaluated using a fitness function[25].

The PSO algorithm determines the direction and speed of each particle's travel using a velocity vector. Each particle's velocity vector is updated depending on two factors: its previous velocity and the difference between its present position and its best position so far. This allows particles to migrate toward better solutions while avoiding becoming stuck in local optima.

Two fundamental parameters define the PSO algorithm: the number of particles and the inertia weight. The size of the swarm is determined by the number of particles, while the inertia weight governs the balance between global and local search. A greater inertia weight favors global search, whereas a lower inertia weight favors local search[26].

PSO parameters

The following are the primary parameters used to model the PSO:

1. $S(n) = \{s_1, s_2, \ldots, s_n\}$: a swarm of n particles
2. s_i: a swarm individual with position p_i and velocity $v_i, iin[|1, n|]$
3. p_i: a particle's location s_i
4. v_i: a particle's velocity p_i

5. $pbest_i$: a particle's best solution

6. $gbest$: the swarm's best answer (Global)

7. f which is the fitness function.

8. c_1, c_2: Acceleration constants

9. r_1, r_2: random numbers between 0 and 1

10. t: the number of iterations[27]

PSO Algorithm Details: Mathematical Models

The PSO algorithm relies on two primary equations. The first equation (3.1) is the velocity equation, in which each particle in the swarm modifies its velocity based on its current position, the computed values of the individual and global best solutions, and other factors. The acceleration factors associated with the personal and social components are denoted by the coefficients c_1 and c_2. They are referred to as trust parameters because c_1 models a particle's confidence in itself, whereas c_2 models a particle's confidence in its neighbors [27]. They define the stochastic impact of thinking and social behavior along with the random numbers r_1 and r_2.

$$v_i^{t+1} = V_i^t + c_1 r_1 (pbest_i^t - p_i^t) + c_2 r_2 (gbest^t - p_i^t) \tag{3.1}$$

The second equation (3.2) is the position equation, where each particle updates its position using the newly calculated velocity:

$$p_i^{t+1} = p_i^t + V_i^{t+1} \tag{3.2}$$

The position and velocity parameters are co-dependent, which means that the velocity depends on the location and vice versa[28].

PSO execution steps are as follows:

1. Initialize the algorithm constants.

2. Initialize the solution from the solution space (Initial values for location and velocity).

3. Evaluate the fitness of each particle.

4. Updated individual and global bests (pbest and gbest)

5. Update the velocity and position of each particle.

6. Repeat step 3 until the termination condition is met[27].

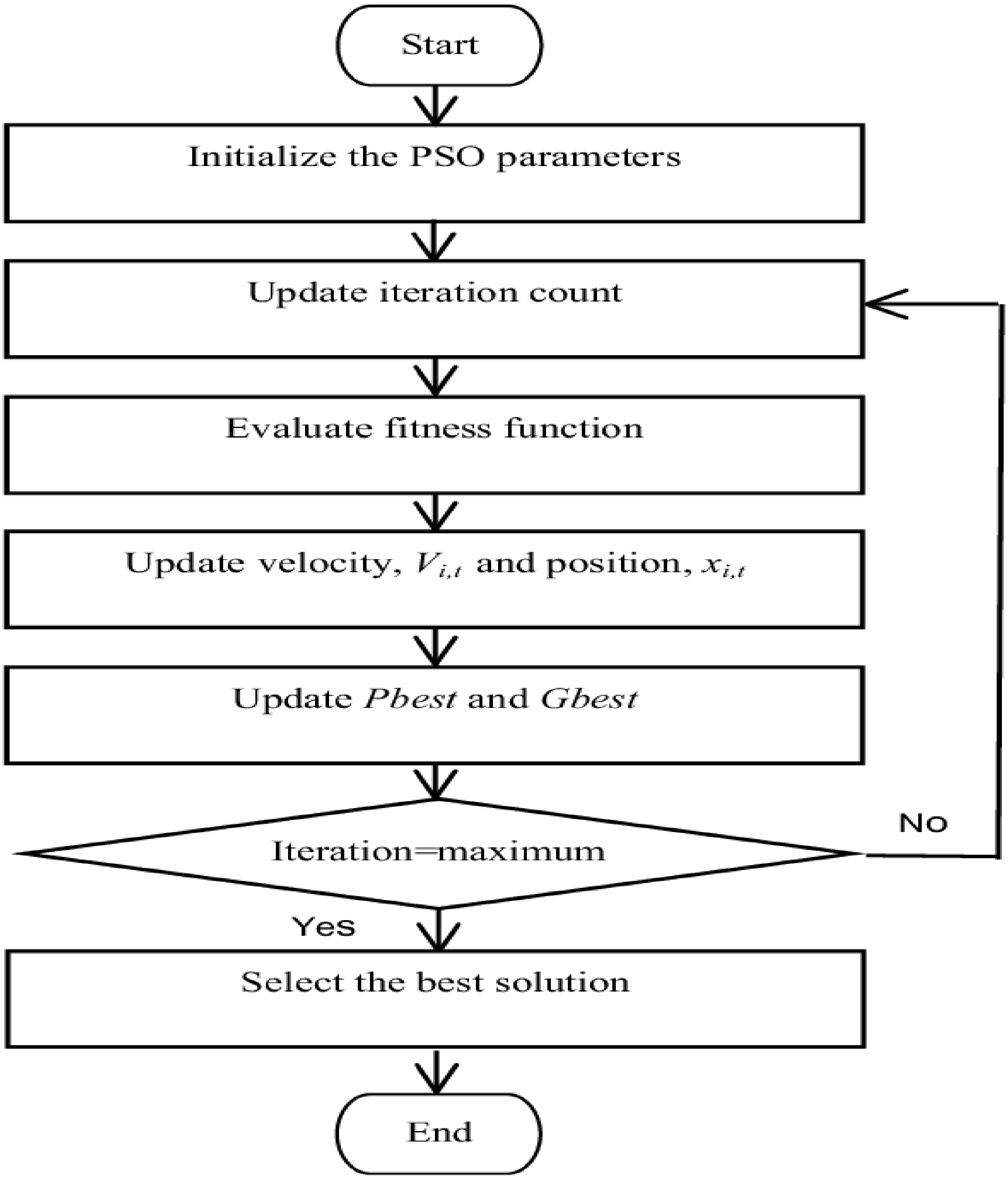
Start
Initialize the PSO parameters
Update iteration count
Evaluate fitness function
Update velocity, $V_{i,t}$ and position, $x_{i,t}$
Update $Pbest$ and $Gbest$
Iteration=maximum
No
Yes
Select the best solution
End

Figure 3.3: PSO Algorithm Flowchart
[29]

PSO has been used to solve a variety of optimization problems, including engineering design, scheduling, and data mining. Its benefits include its simplicity, versatility, and capacity to quickly converge on a worldwide solution. However, it is susceptible to premature convergence, and its performance is greatly dependent on parameter selection and problem characteristics[30].

3.2.2 Firefly Algorithm

The Firefly algorithm (FA) is a metaheuristic optimization technique inspired by firefly flashing. Xin-She Yang proposed the algorithm for the first time in 2008[31].

The Firefly Algorithm simulates firefly flashing behavior to optimize a given objective function. Each firefly in the algorithm represents a potential solution to the problem being solved. A brightness parameter proportional to the objective function value of the related solution models the flashing behavior of a firefly. A firefly's brightness attracts other fireflies, and the attractiveness reduces as the distance between the fireflies increases[32].

Ruless of Firefly Algorithm

Before diving into the rules, keep in mind that all fireflies are unisex, which means that one firefly will be attracted to another firefly regardless of sex and that the brightness of the firefly is determined by the objective function.

The Firefly Algorithm (FA) models fireflies to exhibit the following behavior and rules:

1. Attractiveness: Fireflies are drawn to the light of other fireflies. A firefly's brightness is proportional to the objective function value. Greater brightness implies improved fitness.

2. Movement: Fireflies move at random, however, they prefer brighter fireflies. The stronger the attraction, the brighter the firefly.

3. Intensity The brightness of a firefly is determined by its intensity. The intensity reduces as one moves away

4. Flashing: Fireflies can change the frequency of their flashing to attract others. The strength of the attraction is determined by the frequency of the flashing[33].

Firefly Algorithm Details: Mathematical Models

The cost function of the given problem determines the firefly's light intensity. The firefly algorithm follows the equation below.

The light intensity reduction abides by the law of inverse square(see (3.3)):

$$I(r) = \frac{I_s}{R^2} \tag{3.3}$$

Where $I(r)$ denotes the intensity at the source and r is the distance from the source

If we consider the absorption coefficient γ, the light intensity I vary with the square distance r.

$$I(r) = I_0 exp^{-\gamma r^2} \tag{3.4}$$

Where I_0 is the initial value at (r = 0) and Equation (3.5) expresses the attractiveness, where β_0 is the beginning value at $(r = 0)$:

$$\beta = \beta_0 e^{-\gamma r^2} \tag{3.5}$$

Equation (3.6) calculates the Cartesian distance between two fireflies, i and j, at their respective coordinates xi and x_j. Where D indicates the problem's dimensionality and x_{ik} is the k^{th} element of spatial coordinate x_j of the i^{th} firefly[34].

$$|r_{i,j}| = |r_i - r_j| = \sqrt{\sum_{k=1}^{D}(x_{i,k} - x_{j,k})^2} \tag{3.6}$$

Finally, equation (3.7) determines the movement of a firefly i^{th} when it is attracted to another (j^{th}) attractive (brighter) firefly [34].

$$x_i(t+1) = x_i(t) + \beta_0 e^{-\gamma r^2}(x_j(t) - x_i(t)) + \alpha\epsilon \tag{3.7}$$

where x_i(t + 1) is the firefly i position at iteration $t + 1$ displacement. As can be observed, the first element on the right side of Equation (3.7) is the position of firefly i at iteration t, the second is relative to attraction, and the final is randomization (blind flying if no light) where α is the random walk parameter $[0, 1)$ [34].

Firefly Algorithm Execution steps and Flowchart

Execution steps:

1. Initialize Parameters (Population N, γ, β, α)
2. Generate initial population of n flies
3. Compute fitness (intensity) for each firefly member
4. Check if ($t := 1$ to max iteration)
5. Update the position and fitness (light Intensity) of each firefly
6. Return The Best Solution [35]

The FA algorithm iteratively optimizes a given objective function by changing the positions of fireflies in the search space according to these behaviors and rules. The algorithm begins by randomly initializing a swarm of fireflies and then changes each firefly's position based on its attraction to other fireflies in the swarm. The process is repeated until a stopping requirement (e.g., a maximum number of iterations is achieved or a good solution is found) is met. Figure 3.4 (see below) shows the Flow chart of the Firefly Algorithm.

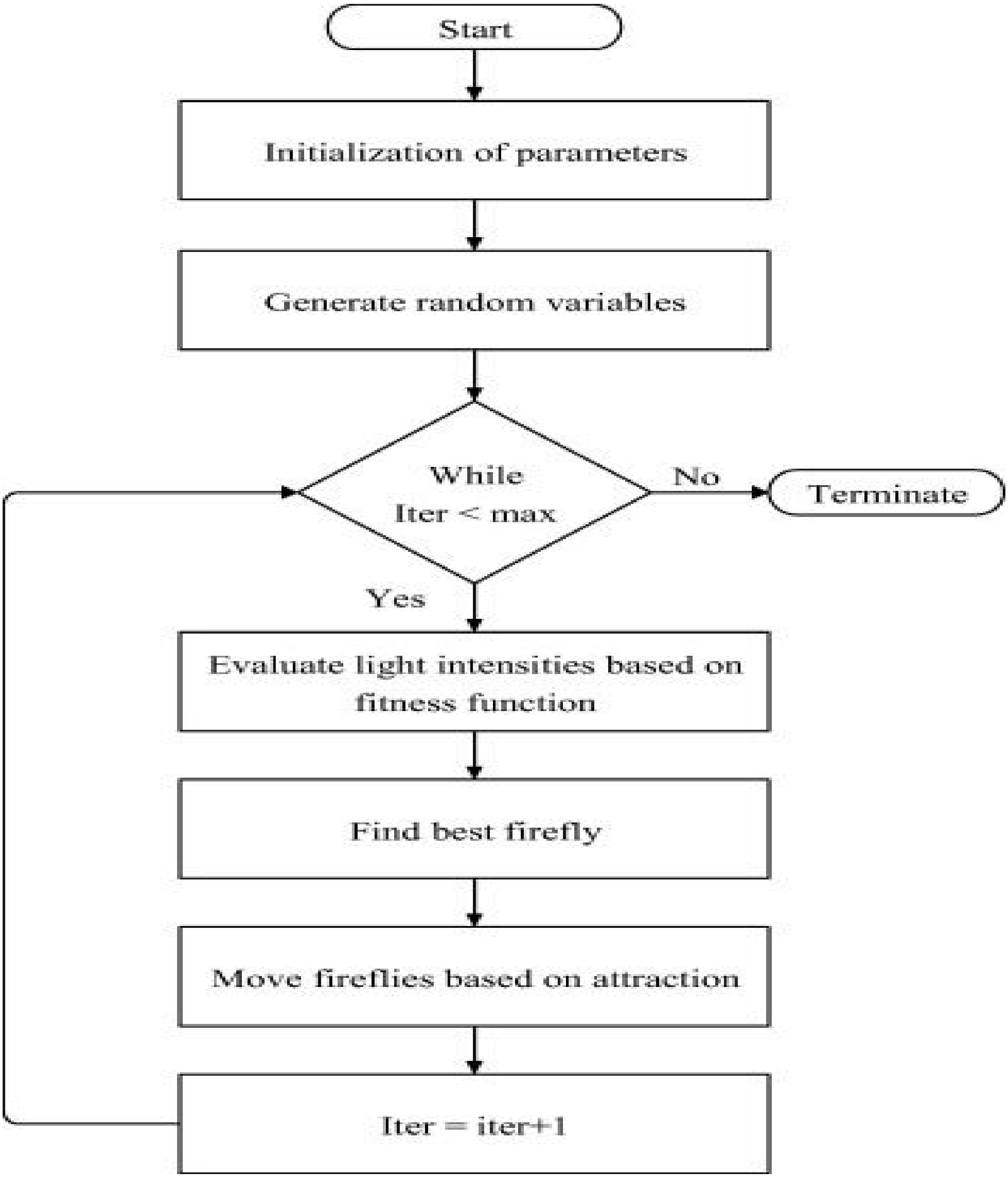
Start
Initialization of parameters
Generate random variables
While
Iter < max
No
Terminate
Yes
Evaluate light intensities based on fitness function
Find best firefly
Move fireflies based on attraction
Iter = iter+1

Figure 3.4: Firefly Algorithm Flowchart
[36]

The Firefly Algorithm has been used to solve a wide variety of optimization issues in industries such as engineering, finance, and healthcare. It has been demonstrated to be effective in discovering high-quality solutions and has piqued the interest of the optimization research community.

3.3 Evaluation metrics: Classification Report

A classification report is a standard machine-learning tool for assessing the performance of a classification model[37]. It presents a detailed summary of several metrics including as precision, recall, F1-score, and support, which are used to evaluate the model's performance in predicting the correct class labels for a given set of data .

1. Precision is a ratio of accurate predictions, mostly known as True positive (TP) to the total of both accurate(TP) and false positive(FP) predictions. It evaluates how accurately the model made predictions that turned out to be true. Less false positives are indicated by increased precision(see (3.8).

2. Recall: The ratio of true positive predictions to the total of true positive and false negative (FN) predictions is known as sensitivity, also known as sensitivity or true positive rate (TPR). It assesses the model's capacity to distinguish accurately between false positives and true positives in the data. Less false negatives are indicated by a stronger recall(see (3.9).

3. F1-score: The harmonic mean of recall and precision is known as the F1-score. It strikes an acceptable balance between recall and precision and offers a single value that measures the model's overall accuracy. It is a frequently used metric for assessing how well a model performs(see (3.10).

4. Support: The amount of instances of each class in the actual data is known as support. It is helpful for interpreting the model's performance and offers details on the distribution of classes in the dataset [38].

These metrics' calculations are based on counts of predictions that the classification model made that were true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN). The following formulas are used to determine the precision, recall, and F1-score:

$$Preciscion = \frac{TP}{TP + FP} \tag{3.8}$$

$$Recal = \frac{TP}{TP + FN} \tag{3.9}$$

$$F1 - score = \frac{2 \cdot Preciscion \cdot Recall}{Precision + Recall} \tag{3.10}$$

Lastly, the count of instances of each class in the real data is used to calculate support.

It's worth noting that these measures are used in combination to assess the overall effectiveness of a classification model. A high precision shows that the model makes accurate positive predictions, a high recall suggests that the model captures the majority of true positive instances, and a high F1-score indicates a good balance of precision and recall. It's also vital to evaluate the support because unequal class distributions can have an impact on the performance of these metrics[39].

Chapter 4

Experiments and Results

Chapter 5

Conclusion

Bibliography

[1] Justin Waring, Charlotta Lindvall, and Renato Umeton. “Automated machine learning: Review of the state-of-the-art and opportunities for healthcare”. In: *Artificial Intelligence in Medicine* 104 (2020), p. 101822. DOI: 10.1016/j.artmed.2020.101822.

[2] Frederico H C Ara’ujo, Andr’e M Santana, and Pedro de A. Santos Neto. “Using machine learning to support healthcare professionals in making preauthorisation decisions”. In: *International Journal of Medical Informatics* 94 (2016), pp. 1–7. DOI: 10.1016/j.ijmedinf.2016.06.007.

[3] Pitt SHRS Online. *The Role of Data Analytics in Health Care*. https://online.shrs.pitt.edu/blog/data-analytics-in-health-care/. [Online; accessed 26-April-2023]. 2021.

[4] Yu Liu et al. “A comparison of XGBoost, Random Forest, LSSVM and SVM based on three medical datasets”. In: *International Journal of Online and Biomedical Engineering* 15.11 (2019), pp. 48–66. DOI: 10.3991/ijoe.v15i11.10390.

[5] Tianqi Chen and Carlos Guestrin. “XGBoost: A Scalable Tree Boosting System”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016). DOI: 10.1145/2939672.2939785.

[6] Xin-She Yang and Mehmet Karamanoglu. “Swarm Intelligence and Bio-Inspired Computation: An Overview”. In: *Swarm Intelligence and Bio-Inspired Computation*. Elsevier, 2013, pp. 3–23. DOI: 10.1016/B978-0-12-405163-8.00001-6.

[7] Filipe da Silva Gonçalves. “Predictive analysis in healthcare”. MA thesis. Lisbon, Portugal: Iscte - Instituto Universit’ario de Lisboa, 2018.

[8] Basma Boukenze, Hajar Mousannif, and Abdelkrim Haqiq. "Predictive Analytics in Healthcare System Using Data Mining Techniques". In: vol. 6. Apr. 2016, pp. 01–09. DOI: 10.5121/csit.2016.60501.

[9] Jie Sheng et al. "Predictive Analytics for Care and Management of Patients With Acute Diseases: Deep Learning-Based Method to Predict Crucial Complication Phenotypes". In: *Journal of Medical Internet Research* 23.2 (2021), e18372.

[10] N Divyashree and Nandini Prasad. "Improved Clinical Diagnosis Using Predictive Analytics". In: *IEEE Access* 10 (Jan. 2022), pp. 1–1. DOI: 10.1109/ACCESS.2022.3190416.

[11] Ali Haidar, Brijesh Verma, and Rim Haidar. "A Swarm-based Optimization of the XGBoost Parameters". In: *arXiv preprint arXiv:2101.02175* (2021).

[12] Sotiris B. Kotsiantis et al. "XGBoost Optimized by Adaptive Particle Swarm Optimization for Credit Scoring". In: *Mathematical Problems in Engineering* 2021 (2021), p. 6655510. ISSN: 1024-123X. DOI: 10.1155/2021/6655510. URL: https://doi.org/10.1155/2021/6655510.

[13] Marija Zivkovic et al. "Novel hybrid firefly algorithm: An application to enhance XGBoost tuning for intrusion detection classification". In: *PeerJ Computer Science* 8 (2022). DOI: 10.7717/peerj-cs.956.

[14] Antonio J Peña et al. "Cervical Cancer Diagnosis Model Using Extreme Gradient Boosting and Bioinspired Firefly Optimization". In: *Scientific Programming* 2021 (2021), p. 5540024. DOI: 10.1155/2021/5540024.

[15] Aleksandar Petrovic et al. "Intrusion Detection by XGBoost Model Tuned by Improved Multi-verse Optimizer". In: *International Conference on Applied Physics, Simulation, and Computing*. Springer. 2023.

[16] IBM. "What is machine learning?" In: (2021). Accessed: May 1, 2023. URL: https://www.ibm.com/topics/machine-learning.

[17] JavaTpoint. *Supervised Machine Learning*. 2021. URL: https://www.javatpoint.com/supervised-machine-learning (visited on 05/02/2023).

[18] Daniel Johnson. "Unsupervised Machine Learning: Algorithms, Types with Example". In: (2023). URL: https://www.guru99.com/unsupervised-machine-learning.html.

[19] Yulia Gavrilova. *How to Choose a Machine Learning Technique*. July 8th, 2020. URL: https://serokell.io/blog/how-to-choose-ml-technique (visited on 05/02/2023).

[20] Avik$_D$*utta*. *Ensemble Classifier | Data Mining*. 2023-05-02. URL: https://www.geeksforgeeks.org/ensemble-classifier-data-mining/ (visited on 05/02/2023).

[21] T. Chen and Guestrin. "XGBoost: A Scalable Tree Boosting System." In: *XGBoost* (). URL: https://doi.org/10.1145/2939672.2939785.

[22] Smartdraw. "Decision Tree". In: *J. Object Oriented Program.* 1.1 (Jan. 2023). URL: https://www.smartdraw.com/decision-tree/.

[23] Jason Brownlee. "A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning". In: *A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning* (). URL: https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/.

[24] Yuhui Shi and Russell Eberhart. "A Modified Particle Swarm Optimizer". In: *Proceedings of the IEEE International Conference on Evolutionary Computation*. 1998, pp. 69–73. DOI: 10.1109/ICEC.1998.699146.

[25] Daniel Bratton and James Kennedy. "Defining a Standard for Particle Swarm Optimization". In: *Proceedings of the IEEE Swarm Intelligence Symposium*. 2007, pp. 120–127. DOI: 10.1109/SIS.2007.368705.

[26] Andries P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. Wiley, 2005.

[27] Baeldung. *How Does Particle Swarm*. URL: https://www.baeldung.com/cs/pso#3-mathematical-models.

[28] Adrian Tam. *A Gentle Introduction to Particle Swarm Optimization*. URL: https://machinelearningmastery.com/a-gentle-introduction-to-particle-swarm-optimization/.

[29] Zetty N. Zakaria et al. "An Extension of Particle Swarm Optimization (E-PSO) Algorithm for Solving Economic Dispatch Problem". In: *2013 1st International Conference on Artificial Intelligence, Modelling and Simulation* (2013), pp. 157–161.

[30] Russell C. Eberhart and James Kennedy. "A New Optimizer Using Particle Swarm Theory". In: *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. 1995, pp. 39–43. DOI: 10.1109/MHS.1995.494215.

[31] Xin-She Yang. "Firefly algorithms for multimodal optimization". In: *In: Stochastic Algorithms: Foundations and Applications, SAGA 2009, Berlin, Heidelberg* (2008), pp. 169–178. DOI: 10.1007/978-3-540-87700-4_14.

[32] Xin-She Yang. "Firefly algorithm, stochastic test functions and design optimisation". In: *International Conference on Modelling, Simulation and Optimization*. 2010, pp. 81–89. DOI: 10.1109/ICMSO.2010.5544696.

[33] Seyedali Mirjalili and Seyed Mohammad Mirjalili. "How effective is the Grey Wolf optimizer in training multi-layer perceptrons". In: *Applied Intelligence* 43.6 (2015), pp. 1503–1517. DOI: 10.1007/s10489-015-0718-1.

[34] Smail Bazi et al. "A Fast Firefly Algorithm for Function Optimization: Application to the Control of BLDC Motor". In: *Sensors* 21.16 (2021). ISSN: 1424-8220. DOI: 10.3390/s21165267. URL: https://www.mdpi.com/1424-8220/21/16/5267.

[35] Sabri Arik et al. "Self-Adaptive Step Firefly Algorithm". In: *Journal of Applied Mathematics* 2013 (2013), p. 832718. ISSN: 1110-757X. DOI: 10.1155/2013/832718. URL: https://doi.org/10.1155/2013/832718.

[36] P. Balachennaiah, M. Suryakalavathi, and P. Nagendra. "Firefly algorithm based solution to minimize the real power loss in a power system". In: *Ain Shams Engineering Journal* 9.1 (2018), pp. 89–100. DOI: 10.1016/j.asej.2015.10.005.

[37] F. Pedregosa et al. *Scikit-learn: Machine Learning in Python*. https://scikit-learn.org/stable/index.html. 2011.

[38] M. Vlasenko. *Understanding Classification Report in Scikit-learn*. 2020. URL: https://towardsdatascience.com/understanding-classification-report-in-sklearn-a9c85db76e95.

[39] J. Brownlee. *Classification Report Explained for Machine Learning in Python*. 2019. URL: https://machinelearningmastery.com/classification-report-explained-for-machine-learning-in-python/.